U.S.P.S. Express Mail Label No.: EL 848969294 US

Date of Deposit: August 22, 2003

ATTORNEY DOCKET NO. 14911US02

UPDATE PACKAGE GENERATOR EMPLOYING PARTIAL PREDICTIVE MAPPING TECHNIQUES FOR GENERATING UPDATE PACKAGES FOR MOBILE HANDSETS

CROSS-REFERENCE TO RELATED APPLICATIONS/INCORPORATION BY REFERENCE

[01] This patent application makes reference to, claims priority to and claims benefit from United States Provisional Patent Application Serial No. 60/405,253, entitled "Firmware Update Network And Process Employing Preprocessing Techniques," filed on August 22, 2002, United States Provisional Patent Application Serial No. 60/415,620, entitled "System for Generating Efficient And Compact Update Packages," filed on October 2, 2002, United States Provisional Patent Application Serial No. 60/441,867, entitled "Mobile Handset Update Package Generator That Employs Nodes Technique," filed on January 22, 2003, and United States Provisional Patent Application Serial No. 60/447,977, entitled "Update Package Generator Employing Partial Predictive Mapping Techniques For Generating Update Packages For Mobile Handsets," filed on February 18, 2003.

[02] The complete subject matter of each of the above-referenced United States Patent Applications is hereby incorporated herein by reference, in its entirety. In addition, this application makes reference to United States Provisional Patent Application Serial No. 60/249,606, entitled "System and Method for Updating and Distributing Information," filed November 17, 2000, and International Patent Application Publication No. WO 02/41147 A1, entitled "Systems And Methods For Updating And Distributing Information," publication date March 23, 2002, the complete subject matter of each of which is hereby incorporated herein by reference, in its entirety.

[03] This application is also related to the following co-pending applications, the complete subject matter of each of which is hereby incorporated herein by reference in its entirety:

Ser. No.	Docket No.	Title	Filed	Inventors
	14121US02	Firmware Update Network and Process Employing Preprocessing Techniques	August 21, 2003	Chen Gustafson
	14122US02	System for Generating Efficient and Compact Update Packages	August 21, 2003	Chen Gustafson Barber
	14312US02	Mobile Handset Update Package Generator That Employs Nodes Technique	August 21, 2003	Chen

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[04] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[05] [Not Applicable]

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[06] The present invention relates generally to update of firmware/software components in mobile handsets using an update agent, and, more specifically, to the generation of compact update packages employing optimization techniques.

BACKGROUND OF THE ART

[07] Electronic devices, such as mobile phones and personal digital assistants (PDAs), often contain firmware and application software that are either provided by the manufacturer of the electronic devices, by telecommunication carriers, or by third parties. These firmware and application software often contain software bugs. New versions of the firmware/ software are periodically released to fix the bugs or to introduce new features, or both.

[08] When informing a mobile electronic device of the need to update its firmware/software, problems might arise. For example, problems may arise when attempting to control the size of an update package containing the difference information. Update package generation is prone to an "avalanche" effect, wherein a minor change to code may result in relocation of a large amount of other code or components. The relocation of the code may result in difference information that is very large, thereby prolonging the download process and making it more costly. Other problems may arise, such as the inability to clearly and efficiently specify the steps needed to update firmware/software in the electronic device to a new version.

[09] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[10] Aspects of the present invention may be seen in a mobile services network comprising a mobile electronic device having a non-volatile memory, a random access memory and security services; the network further comprising a management server; an update package repository; and a generator with a partial predictive mapping (PPM) preprocessor for generating update packages for updating an old version of a firmware to a new version of the firmware. The update packages generated by the generator with a partial predictive mapping (PPM) preprocessor may incorporate a shift region list.

[11] The process of generating an update package with the generator with a partial predictive mapping (PPM) preprocessor comprises creating a module map of module locations and sizes in the old image of firmware versus the new image of firmware; creating a shift region list; and generating an update package with the shift region list information.

[12] The shift region list is created by, for example, identifying shift points within each module of the firmware, wherein the shift points define shift regions; creating and modifying a first shift region list to include external shifts; consolidating shift regions across modules if address adjustments remain the same across a module boundary; and creating a second shift region list by consolidating shift regions from the first shift region list.

[13] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[14] Fig. 1 illustrates a block diagram of an exemplary mobile services network, in accordance with an embodiment of the present invention.

[15] Fig. 2 illustrates an example of a new version of code where the modules have shifted but are in the same order as the old version of code, in accordance with an embodiment of the present invention.

[16] Fig. 3 illustrates an example of a new version of code where the modules are not in the same order as the old version of code, in accordance with an embodiment of the present invention.

[17] Fig. 4 illustrates an example of inserting shift points to define shift regions as applied to Fig. 3, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[18] Fig. 1 illustrates a block diagram of an exemplary mobile services network 105, in accordance with an embodiment of the present invention. In the exemplary embodiment of Fig. 1, the mobile services network 105 comprises a mobile handset 107, a management server 109, an update package repository 133, and a generator with a partial predicting mapping (PPM) preprocessor 145. The mobile handset 107 may have access to services that may include a firmware/software update service. The mobile handset 107 may retrieve update packages with an associated shift region list from the management server 109. The update packages may be generated by the generator with a partial predicting mapping preprocessor 145 and populated into the update package repository 133, which is linked to the management server 109. The update package, generated by the generator with a partial predicting mapping preprocessor 145, may incorporate a shift region list with code-shift-related information within the same package. In one embodiment, the shift region list may comprise a list of a two-byte shift region number, a three-byte shift region length, and a four-byte adjustment value, as shown in an example hereinafter. In another embodiment of the present invention, the mobile handset 107 may be a personal digital assistant (PDA) or any mobile electronic device.

[19] In one embodiment, the generator with a partial predicting mapping preprocessor 145 may be coupled to the update package repository 133. The management server 109 may access the update package repository 133 to retrieve the update packages and associated information, such as metadata, shift region list information, etc.

[20] In another embodiment, the generator with a partial predicting mapping preprocessor 145 may be located at a location remote from the update package repository 133. In such an embodiment, the output of the generator with a partial predicting mapping preprocessor 145, namely update packages and associated shift region list information, etc., may be communicated to the update package repository 133 through a storage medium, such as a CD-ROM disk, and loaded into the update package repository 133. Then, the update package may be disseminated to the mobile handset 107 via the management server 109.

The generator with a partial predictive mapping preprocessor 145 may generate an update package and associated information, by comparing two different versions of the firmware/software of the mobile handset 107.

[21] The mobile handset 107 may comprise a non-volatile memory 111, a random access memory (RAM) 125, and security services 123. The non-volatile memory 111 of the mobile handset 107 may comprise an update agent 113 with partial predictive mapping support, a firmware and real-time operating system (RTOS) 115, an operating system (OS) layer 117, a download agent or browser 119, an end-user-related data and content 121, and boot initialization 127.

[22] In one embodiment, the mobile handset 107 downloads an update package from the update package repository 133, and then reboots. Upon rebooting, the mobile handset 107 starts up, executes the boot initialization 127, and determines whether the update agent 113 needs to execute the update process. The decision to execute the update process by the update agent 113 may be based on status information such as, for example, the availability of an update package. The status information may be available in the non-volatile memory 111. If it is determined that the update agent 113 needs to execute the update process, the mobile handset 107 may invoke the update agent 113.

[23] In one embodiment, the generator with a partial predictive mapping preprocessor 145 may generate a list of shift regions when it generates an update package. Each entry in the list of shift regions may comprise three values, for example, one value indicating a shift region number, another value indicating shift region length, and the last value indicating an adjustment value used as an offset, as illustrated in an example hereinafter. The shift region list may be incorporated into the update package and loaded into the update package repository 133. When the mobile handset 107 requests an update package from the management server 109, the update package may be delivered to the mobile handset 107 from the update package repository 133 via the management server 109. The update agent 113 in the mobile handset 107 may execute a "preprocessing" step using the shift region list, to prepare the existing or old image of the firmware/software for update. After the

preprocessing, the update agent 113 may employ other information from the update package to update the firmware/software.

[24] In one embodiment, the partial predictive mapping technique for preprocessing binary images of firmware/software may utilize "logical" alignment elements within different versions of code such as, for example, an old version and a new version of firmware. The output of the partial predictive mapping preprocessor component of the generator with a partial predictive mapping preprocessor 145 may include shift-related information in terms of location of the region, length of the region, and the associated offset. The shift-related information may be incorporated into the update package to be sent as a single package to the mobile handset 107.

[25] The update package generation by the generator with a partial predictive mapping preprocessor 145 may be used for mobile handsets and other electronic devices, such as devices that contain code or data in non-volatile memory which may require updates to newer or different versions.

[26] Fig. 2 illustrates an example of a new version 201 of code where the modules have shifted but are in the same order as the old version 203 of code, in accordance with an embodiment of the present invention. Preprocessing techniques may be utilized to attempt aligning of the elements of an old version of a code, such as firmware or software, with elements of the new version of the code. In one embodiment, the process of aligning code may handle shifts in module locations, and predict the shift values of branch instruction parameters (and any similar address-based or offset-based instructions) in the binary firmware image. The preprocessing methods described herein have proven highly efficient in situations as shown in Fig. 2, where the old version 203 and the new version 201 look relatively similar, in that the modules within each version remain in the same order relative to each other, but have shifted in location.

[27] Fig. 3 illustrates an example of a new version 301 of code where the modules are not in the same order as the old version 303 of code, in accordance with an embodiment of the present invention. In such a case, partial predictive mapping may be used. Partial predictive

mapping identifies shifts in module locations as well as "shift points" within modules, and uses this information to efficiently encode a list of "shift regions." The shift region list may then be used to adjust branch link address parameters within the region, without explicitly mapping each branch link instruction in the old image to its counterpart in the new image.

[28] In a firmware image, there may be thousands of modules. Each module may shift its location and/or change its order relative to other modules, as show in Fig. 3. Additionally, each module may change internally, as shown in Fig. 3 with respect to module 1, where it grew from a smaller size module 1, 315, in the old image 303 to a larger size module 1, 305, in the new image 301. These changes can be described in terms of "shift regions" bounded by "shift points." A "shift region" is a contiguous range of code in which branch link target address/offset parameters change by a constant amount for target addresses that are "internal" (an address in the shift region). Partial predictive mapping may also handle branch link target addresses that are "external" (a different shift region than the one in which the branch link instruction resides).

[29] In one embodiment, as illustrated by the example of Fig. 3, module 4 (321 and 311) and module 5 (323 and 313) did not change order, but were shifted forward as a result of module 1 (315 and 305) growing in size. Therefore, there is a shift point at the beginning of module 4 (321 and 311) denoting the beginning of a shift region that comprises all of module 4 (321 and 311) and module 5 (323 and 313).

[30] In the example illustrated by Fig. 3, the addition to module 1 (315 and 305) may be an insertion of code, data, etc., somewhere in the middle of the old module version, with everything else in the module remaining the same. This insertion causes "internal" branch link address/offset parameters after the addition to shift forward by a constant amount within each shift region. Therefore, partial predictive mapping defines a shift point in version 1 (303) where an insertion is detected, denoting a shift region from that point to the end of module 2 (317).

[31] Fig. 4 illustrates the shift points and shift regions discussed hereinabove as applied to the examples illustrated by Fig. 3. A shift point (425) is defined at the beginning of the old

image 403, or at the start of module 1 (415). This shift point (425) defines a shift region equal to module 1 (415) in the old image 403. A second shift region equal to module 2 (417) is defined by a shift point (427) at the start of module 2 (417). A third shift region equal to module 3 (419) is defined by a shift point (429) at the start of module 3 (419). A fourth shift region equal to modules 4 and 5 (421 and 423) is defined by a shift point (431) at the start of module 4 (421). As a result of inserting the shift points, there are a total of four shift regions within the new version (401) of firmware. In the example illustrated by Fig. 4, in the old image (403) the defined shift regions are defined by module boundaries, in other embodiments shift regions may not correspond to module boundaries. The mapping of the shift regions between the old image (403) and the new image (401) compares the positions of the defined shift regions in term of the shift points defining their starting points in the old image (403) with their starting points in the new image (402). This will be explained further in an example hereinafter.

[32] A partial predictive mapping shift region list is a list of shift regions as described hereinabove, along with their shift adjustments. A shift region list may be created by a generator analysis module, and used by a preprocessor in a generator with a partial predictive mapping preprocessor to modify branch link address/offset parameters throughout the affected portion of the firmware image. The modifications may adjust branch link address/offset parameters in the old image of the firmware to match the new image of the firmware. The generator with a partial predictive mapping preprocessor may then process the modified old image and the new image to produce difference information comprising core generator instructions. Using this approach, the partial predictive mapping preprocessor may not need to produce accurate adjustments to branch link address parameters in all cases, but improves core generator efficiency in the majority of cases. Hence, partial predictive mapping is designed to produce a high accuracy of branch link address parameter adjustment despite module shifts and/or changes in module order.

[33] In one embodiment, the partial predictive mapping implementation may utilize executable and linking format (ELF) files to determine module location changes. However, shift region lists may also be created by using only the old and new binary images of the

firmware, such that partial predictive mapping may change from a tool chain-specific approach to a CPU-specific approach. The preprocessor may determine where modules begin and end in both the old and the new images of firmware, by analyzing the old and new binary images.

[34] To create a shift region list, the partial predictive mapping preprocessor may identify shift points. In one embodiment, the partial predictive mapping preprocessor may identify shift points within each module, and consolidate shift regions across modules if branch link adjustments are the same across a module boundary. This may be done, for example, using two preprocessing operations, one for analysis, and another for preprocessing shift information to be output to the core generator.

[35] The analysis operation may consist of several steps. A first step may create a module map, which may be comprised of module locations and sizes in the old image of firmware and the new image of firmware. The module map may be used by the partial predictive mapping preprocessor in the generator with a partial predictive mapping preprocessor. The next step, step 2, in creating a shift region list may create an initial shift region list for "internal" shifts. In this list, each shift region may correspond to a module in the old image of firmware. The "adjustment" value in this case may be the difference between the start location of the module in the old image and the start location of the same module in the new image of firmware. Step 3 may modify the shift region list for "external" shifts. In this step, the preprocessor may identify the modules that have changed in size from the old image to the new image of firmware. The preprocessor may then adjust the branch link addresses in the old image of firmware by the adjustment values from step 2. Adjusting the branch link addresses makes it possible to identify areas where new code may have been inserted, added, or removed. Each area of the new image that is different from the old image may result in a shift point inside the module being analyzed. These shift points may form the boundaries for shift regions inside that module in the old image of firmware. At this point in the process, the existing shift region list entries for each module that was identified as changing size may be deleted, and the preprocessor may insert the appropriate entries for the newly identified shift regions into the shift region list. Proceeding from the beginning of the shift region list,

contiguous shift regions having the same adjustment values may be identified. Such entries may be consolidated into one shift region. The module map may then be discarded. This process produces a region shift list, which may be sent with a standard update package. A shift region list produced in this manner may be utilized by the partial predictive mapping preprocessor of the update agent to modify the old image of firmware to look more like the new image of firmware. An update algorithm may then be applied to the old image to produce the new image.

[36] A shift region list may be encoded in one of several ways. In one embodiment, a shift region list may be encoded utilizing a shift region matrix as illustrated by an example in the following table:

Shift Region Number (two bytes)	Shift Region Length (three bytes)	Adjustment (four bytes)
1	10kB	+0kB
2	10kB	+12kB
3	10kB	-8kB
4	20kB	+2kB

Since this is a preprocessor, the shift region length is the length of the shift region in the old image of the firmware. In reference to Fig. 4, let us assume that each module in the old image 403 is 10 kilobytes (kB) in size. Let us also assume that module 1 (415) has increased in size by 2kB to result in a new module 1 (405) in the new image 401. As a result module 1 (405) in the new image 401 is 12kB in size. As the matrix above shows, the starting address or the first shift point 425 of module 1, defining the beginning of shift region 1 remains the same, hence, shift region 1 equal to module 1 (415) in the old image 403 needs no adjustment, or an adjustment value of +0kB. Shift region 2, equivalent to module 2 (417) and defined by shift point 427 in the old image 403, has changed module order with shift

region 3, equivalent to module 3 (419) defined by shift point 429 in the old image 403. The shift points 427 and 429 in the old image 403 are now shift points 437 and 435 in the new image 401, respectively. Hence shift point 427 has moved forward from its original position in the old image 403 by 12kB, which is the size of module 3 (10kB) plus the increase in size of module 1 (2kB) illustrated by the area between the point 433 and the shift point 435. In a similar analysis, shift point 429 has moved backward from its original position in the old image 403 by 8kB, since in the old image 403 shift point 429 was 20kB below the start shift point 425, whereas in the new image 401 the corresponding shift point 435 defining the starting point of module 3 (407) is directly after module 1 (405), hence 12kB from the start of the new image 401. Modules 4 and 5 are contiguous, and have the same adjustment factor, in that they both have been adjusted from their original location, or the original location of the shift point 431 to a new shift point location 439, by the same amount, 2kB, which is the increase in the size of module 1. In this case, module 4 and 5 may be consolidated into one shift region 4 with that adjustment factor, and a length that comprises the total length of modules 4 and 5, which is 20kB.

[37] In one embodiment, the partial predictive mapping algorithm does not simply adjust branch links within a shift region by the adjustment amount stored in the shift region matrix, since that may be incorrect for "external" branch links referencing regions outside that particular shift region. In the example of the shift region matrix shown above, and Fig. 4, if a branch link instruction in shift region 1 references an address within shift region 4, an adjustment of 0kB may not be appropriate, since the bytes in shift region 4 have shifted forward by 2kB. Therefore, the partial predictive mapping preprocessor may distinguish between "internal" branch links that link to addresses within the same shift region, and "external" branch links that branch to addresses within other shift regions. Each branch link may be adjusted according to the adjustment for its referenced ("target") shift region.

[38] Several methods may be used to ensure branch links are adjusted correctly to accommodate any "external" branch links. In one embodiment, the pseudo-code shown below may be used. In this embodiment, it may be assumed that the preprocessor is familiar

with CPU-specific op code and parameter format for branch link and other instructions that use absolute or offset addressing parameters.

```
Set pointer to address zero in old firmware image
Let BeginPoint = 0
For each shift region
        Let EndPoint = current shift region endpoint
        Let LocalAdjustment = current shift region adjustment
        Move pointer to next branch link instruction
        Let TargetAddress = address parameter of branch link instruction
        If TargetAddress is between BeginPoint and EndPoint then
                Modify branch link instruction to adjust address parameter
        Else
                Use shift region list to look up target shift region for that TargetAddress
                Get adjustment for the target shift region (also in shift region list)
                Modify branch link instruction to adjust address parameter
        End If
        Let BeginPoint = EndPoint
Next shift region
```

[39] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.